IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Xinliang David Li	§	Art Unit:	2193
		§		
Serial No.:	10/758,376	§	Confirmation No.:	5436
		§		
Filed:	01/15/2004	§	Examiner:	Insun Kang
		§		
For:	PROGRAM OPTIMIZATION USING OBJECT FILE SUMMARY INFORMATION	§	Atty. Dkt. No.:	200313024-1 (HPC.0743US)

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CORRECTED APPEAL BRIEF PURSUANT TO 37 C.F.R § 41.37

Sir:

The final rejection of claims 1, 2, 4-18, and 20-35 is hereby appealed.

I. REAL PARTY IN INTEREST

The real party in interest is the Hewlett-Packard Development Company, LP. The Hewlett-Packard Development Company, LP, a limited partnership established under the laws of the State of Texas and having a principal place of business at 11445 Compaq Center Drive West, Houston, TX 77070, U.S.A. (hereinafter "HPDC"). HPDC is a Texas limited partnership and is a wholly-owned affiliate of Hewlett-Packard Company, a Delaware Corporation, headquartered in Palo Alto, CA. The general or managing partner of HPDC is HPQ Holdings, LLC.

II. RELATED APPEALS AND INTERFERENCES

None.

III. STATUS OF THE CLAIMS

Claims 1, 2, 4-18, and 20-35 have been finally rejected and are the subject of this appeal. Claims 3 and 19 have been cancelled.

IV. STATUS OF AMENDMENTS

No amendment after the final rejection of January 26, 2009 has been submitted. Therefore, all amendments have been entered.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The following provides a concise explanation of the subject matter defined in each of the independent claims involved in the appeal, referring to the specification by page and line number and to the drawings by reference characters, as required by 37 C.F.R. § 41.37(c)(1)(v). Each element of the claims is identified by a corresponding reference to the specification and drawings where applicable. Note that the citation to passages in the specification and drawings for each claim element does not imply that the limitations from the specification and drawings should be read into the corresponding claim element.

Generally, independent claim 1 recites a method of generating a software program executable binary file, the method comprising:

> accessing a first file (Fig. 1:112) for a first module including source code therein (Spec., p. 4, ln. 15-18; p. 6, ln. 11-24);
>
> accessing a second file (Fig. 1:118, 124) for a second module including object code therein and further including object file summary information (Spec., p. 4, ln. 19-24; p. 5, ln. 1-6; p. 6, ln. 11-24); and
>
> generating the executable binary file from at least the first and second files (Spec., p. 6, ln. 17-24),
>
> wherein the object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table (Spec., p. 6, ln. 25 – p. 7, ln. 6; p. 8, ln. 7-29), and
>
> wherein the object file summary information is used in optimizing the executable binary file generated (Spec., p. 2, ln. 7-8; p. 5, ln. 31 – p. 6, ln. 2).

More specifically, independent claim 1 relates to a method of generating a software program executable binary file. A first file for a first module (specification, page 6, lines 11-24, see files "1.c", "2.c" or "3.c", for example) and a second file for a second module (specification, page page 6, lines 11-24, see files "4.o" or "5.o", for example) are accessed. The first file includes source code (specification, page 6, lines 11-24, see files "1.c", "2.c" or "3.c", for example), and the second file includes object code (specification, page 6, lines 11-24, see files "4.o" or "5.o", for example) and object file summary information (specification, page 6, line 25 through page 7, line 6). An executable binary file is generated from at least the first and second files. (Specification, page 6, lines 17-24.) The object file summary information is used in optimizing the executable binary file generated. (Specification, page 2, lines 7-8, and page 5, line 31 through page 6, line 2.) The object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table. (Specification, page 8, lines 7-29.)

Generally, independent claim 26 recites a system for generating a software program executable file, the system comprising:

> a processing device (Fig. 2:202) configured to execute computer-readable program code;
>
> a memory system (Fig. 2:204) configured to store the computer-readable program code and data;
>
> a source file (Fig. 1:112) for a first module comprising source code stored by the memory system (Spec., p. 4, ln. 15-18; p. 6, ln. 11-24);
>
> an object file (Fig. 1:118, 124) for a second module including computer-readable program code and object file summary information (Spec., p. 4, ln. 19-24; p. 5, ln. 1-6; p. 6, ln. 11-24); and
>
> a translator (Fig. 1:102) comprising computer-readable program code stored by the memory system, wherein the computer-readable program code of the translator is configured to access at least the source and object files and to generate the executable file of the program therefrom (Spec., p. 6, ln. 17-24),
>
> wherein the object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table (Spec., p. 6, ln. 25 – p. 7, ln. 6; p. 8, ln. 7-29), and
>
> wherein the object file summary information is used in optimizing the executable file generated (Spec., p. 2, ln. 7-8; p. 5, ln. 31 – p. 6, ln. 2).

More specifically, independent claim 26 relates to a system for generating a software program executable file. The system includes a processing device (202 in FIG. 2) configured to execute computer-readable program code, and a memory system (204 in FIG. 2) configured to store the computer-readable program code and data. The system also includes a source file (112 in FIG. 1, and see 104 in FIGS. 1 and 2) for a first module comprising source code stored by the memory system (204 in FIG. 2), an object file (see 120 and 122 in FIG. 1) for a second module including computer-readable program code and object file summary information (specification, page 7, lines 25 through page 8, line 3, for example), and a translator (102 in FIGS. 1 and 2) comprising computer-readable program code stored by the memory system (204 in FIG. 2). The computer-readable program code of the translator (102 in FIGS. 1 and 2) is configured to access at least the source and object files and to generate the executable file of the program therefrom

(specification, page 3, lines 1-4, for example). The object file summary information is used in optimizing the executable file generated. (Specification, page 2, lines 7-8, and page 5, line 31 through page 6, line 2). The object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table. (Specification, page 8, lines 7-29.)

Generally, independent claim 31 recites a computer-readable storage medium storing an object file (Fig. 1:118, 124) of a computer programming module, the computer-readable storage medium comprising:

> computer-readable object code for the module (Spec., p. 4, ln. 19-24; p. 5, ln. 1-6); and
>
> computer-readable object file summary information including a summary intermediate representation (SIR) and an extension to a linker symbol table for use by a compiler in optimizing executable code including the module (Spec., p. 6, ln. 11 – p. 7, ln. 6; p. 8, ln. 7-29).

More specifically, independent claim 31 relates to a computer-readable storage medium storing an object file of a computer programming module. The computer-readable storage medium includes computer-readable object code (specification, page 6, lines 11-24, see files "4.o" or "5.o", for example) for the module and computer-readable object file summary information. (Specification, page 2, lines 7-8, and page 5, line 31 through page 6, lines 2.) The computer-readable object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table for use by a compiler in optimizing executable code including the module. (Specification, page 8, lines 7-29.)

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claims 1, 2, 4-14, 16-18, 20, 21, and 24-33 were rejected under 35 U.S.C. § 103(a) as unpatentable over Hiranandani (U.S. Patent No. 5,812,855) in view of Ho (U.S. Patent No. 5,923,882).

B. Claims 22, 23, and 35 were rejected under 35 U.S.C. § 103(a) as unpatentable over Hiranandani in view of Ho and further in view of Lohmann (U.S. Patent No. 5,826,087).

C. Claims 15 and 34 were rejected under 35 U.S.C. § 103(a) as unpatentable over Hiranandani in view of Ho and further in view of Haber (U.S. Patent No. 6,966,055).

VII. ARGUMENT

The claims do not stand or fall together. Instead, Appellant presents separate arguments for various independent and dependent claims. Each of these arguments is separately argued below and presented with separate headings and sub-headings as required by 37 C.F.R. § 41.37(c)(1)(vii).

A. Claims 1, 2, 4-14, 16-18, 20, 21, and 24-33 were rejected under 35 U.S.C. § 103(a) as unpatentable over Hiranandani (U.S. Patent No. 5,812,855) in view of Ho (U.S. Patent No. 5,923,882).

1. Claims 1, 18, 20, 21, 24-26, 28-30.

It is respectfully submitted that the obviousness rejection of claim 1 is erroneous.

To make a determination under 35 U.S.C. § 103, several basic factual inquiries must be performed, including determining the scope and content of the prior art, and ascertaining the differences between the prior art and the claims at issue. *Graham v. John Deere Co.*, 383 U.S. 1, 17, 148 U.S.P.Q. 459 (1965). Moreover, as held by the U.S. Supreme Court, it is important to identify a reason that would have prompted a person of ordinary skill in the art to combine reference teachings in the manner that the claimed invention does. *KSR International Co. v. Teleflex, Inc.*, 127 S. Ct. 1727, 1741, 82 U.S.P.Q.2d 1385 (2007).

Here, it is respectfully submitted that the hypothetical combination of Hiranandani and Ho would not have disclosed or hinted at the claimed subject matter. As conceded by the Examiner, Hiranandani does not disclose object file summary information including an extension to a linker symbol table. 1/26/2009 Office Action at 3. Instead, the Examiner cited Ho as purportedly disclosing the claimed feature. *Id.*

Specifically, claim 1 recites a method of generating a software program executable binary file, the method comprising:

- accessing a first file for a first module including source code therein;
- accessing a second file for a second module including object code therein and further including **object file summary information**; and
- generating the executable binary file from at least the first and second files,
- wherein the **object file summary information** includes a summary intermediate representation (SIR) and an extension to a linker symbol table, and
- wherein the **object file summary information** is used in optimizing the executable binary file generated.

The above concession by the Examiner means that Hiranandani does not disclose that the object file summary information including both a summary intermediate representation (SIR) and an extension to a linker symbol table is used in optimizing the executable binary file that is generated from at least the first file and the second file, where the second file includes object code and the object file summary information.

The Examiner specifically cited the following passage of Ho: col. 4, lines 60-67. The cited passage of Ho refers to a GOT (global offset table), which is explained by Ho as being a table of addresses of all symbols that are referenced. The cited column 4 passage of Ho notes that the content of the GOT is updated by a run-time linker on demand. As further noted by the cited column 4 passage of Ho, by updating the GOT with correct values, a run-time linker can relocate a shared library to a virtual address. A linker relocating a shared library by updating a

GOT, as taught by Ho, is quite different from using object file summary information including a summary intermediate representation and an extension to a linker symbol table to optimize an executable binary file that is generated from first and second files containing source code and object code, respectively, as recited in claim 1.

Ho makes reference to optimizing the GOT 360 by removing indirect addressing – Ho does not describe using the GOT as part of object file summary information that also includes a summary intermediate representation to optimize an executable binary file generated from a first file and a second file, where the second file contains object code and also the object file summary information.

In view of the foregoing, it is respectfully submitted that even if Hiranandani and Ho could be hypothetically combined, the hypothetical combination of references would not have led to the subject matter of claim 1.

Moreover, a person of ordinary skill in the art would not have been prompted to combine the teachings of Hiranandani and Ho. The GOT of Ho is used to address the issue of a main application program not being certain of the exact place in memory where data or instructions of a shared library are stored. Ho, 4:13-22. The issue of a main application program not being sure regarding the exact place in memory of data or instructions of a shared library is not an issue addressed by the summarization information produced by the interprocedural local phase 402a-402n of Hiranandani (as depicted in Fig. 4 of in Fig. 4 of Hiranandani). Hiranandani, 8:26-30. Thus, a person of ordinary skill would not have been prompted to incorporate the GOTs of Ho into the summarization information of Hiranandani, as there simply did not exist any reason to do so.

In view of the foregoing, it is respectfully submitted that the obviousness rejection of claim 1 and its dependent claims is clearly erroneous.

The obviousness rejection of independent claim 26 and its dependent claims is similarly defective.

Reversal of the final rejection of the above claims is respectfully requested.

2. Claims 2, 27.

Claims 2 and 27 depend on respective base claims 1 and 26, and are therefore allowable for at least the same reasons as claims 1 and 26.

Moreover, claim 2 further recites "disambiguating memory accesses otherwise considered aliased using the object file summary information." In the main body of the final Office Action, the Examiner cited column 9, lines 54-59, of Hiranandani as purportedly disclosing this feature of claim 2. The cited column 9 passage of Hiranandani refers to intermediate .o files, and an IPA/IPO phase performing analysis and optimization based on summary information and generating modified or optimized versions of program units in .I files. There is absolutely no mention whatsoever in this passage of Hiranandani regarding **disambiguating memory accesses** otherwise considered aliased using the object file summary information.

The Response to Arguments section of the Office Action further cited to column 12, lines 14-22 of Hiranandani, which refers to the IPA/IPO phase solving interprocedural alias analysis by determining whether a specific procedure can modify a parameter value or not. Determining whether a procedure can modify a parameter value or not in the interprocedural alias analysis of Hiranandani clearly is different from **disambiguating memory** accesses otherwise considered aliased using the object file summary information. The Examiner appears to have merely

identified a keyword, in this case "alias," to make the argument that a passage of Hiranandani containing this keyword is the same as the claimed subject matter. That is clearly not the case.

Claim 2 is therefore further allowable over the cited references for the foregoing reason. Claim 27 is similarly further allowable.

Reversal of the final rejection of the above claims is respectfully requested.

3. Claim 4.

Claim 4 depends from claim 1 and is therefore allowable for at least the same reasons as claim 1.

Moreover, claim 4 further recites that the extension to the linker symbol table includes a flag indicating whether a procedure exposes a memory address by storing the address in a location accessible outside of the procedure. With respect to this element of claim 4, the Examiner cited column 6, lines 44-56, of Ho. 1/26/2009 Office Action at 3-4. The cited passage of Ho refers to collecting additional information about definitions and references of each symbol and attaches this information as attributes to each symbol. The attributes include various elements, but none of these elements constitutes an extension to the linker symbol table including a flag indicating whether a procedure exposes a memory address by storing the address in a location accessible outside the procedure.

The Response to Arguments section of the Office Action further cited column 10, lines 26-47, of Hiranandani. 1/26/2009 Office Action at 13. This passage of Hiranandani refers to a developer's program containing externally visible symbols, such as global variables or subroutines. Nowhere in this passage is there any hint of a flag indicating whether a procedure exposes the memory address by storing the address in a location accessible outside the procedure.

Claim 4 is therefore further allowable for the foregoing reason.

Reversal of the final rejection of the above claim is respectfully requested.

4. Claims 5-14, 16, 17.

Claim 5 depends from base claim 1 and is therefore allowable for at least the same reasons as claim 1. Moreover, claim 5 further recites that the SIR includes a summary symbol table. With respect to claim 5, the Examiner cited column 9, lines 55-59, of Hiranandani. This column 9 passage of Hiranandani refers to a compiler backend not having to be taught to understand summary information. There is absolutely no hint of a summary symbol table in this passage of Hiranandani.

The Response to Arguments section of the Office Action further cited the following passages of Hiranandani as purportedly disclosing the claimed subject matter: column 8, lines 38-57; column 10, lines 26-47. 1/26/2009 Office Action at 13. The cited column 8 passage of Hiranandani refers to an interprocedural local phase generating a second intermediate representation of a source file. The cited column 8 passage also refers to summary information within a program unit that can be used to construct a relationship between inter-program units. However, this passage makes no mention of a summary symbol table.

The cited column 10 passage of Hiranandani refers to linking a DSO file with an object file of a program. The cited column 10 passage also refers to externally visible symbols, such as global variables or subroutines. The cited column 10 passage of Hiranandani also refers to the IPA/IPO phase inputting the DSO file and determining which of the program's externally visible symbols may not be referenced from the outside.

There is also no mention here of a summary symbol table, as recited in claim 5. Claim 5 and its dependent claims are therefore further allowable for the foregoing reason.

Reversal of the final rejection of the above claims is respectfully requested.

5. Claim 31.

Independent claim 31 is also non-obvious over Hiranandani and Ho. With respect to claim 31, the hypothetical combination of Hiranandani and Ho fails to disclose computer-readable object file summary information including a summary intermediate representation and an extension to a linker symbol table for use by a compiler and optimizing executable code including the module. A discussion regarding why the references do not hint at the above claimed subject matter is provided above with respect to claim 1.

Also, for similar reasons as those stated with respect to claim 1, no reason existed that would have prompted a person of ordinary skill in the art to combine the teachings of Hiranandani and Ho to achieve the claimed subject matter. The obviousness rejection of claim 31 and its dependent claims is therefore erroneous.

Reversal of the final rejection of the above claim is respectfully requested.

6. Claims 32, 33.

Claim 32 depends from claim 31 and is therefore allowable for at least the same reasons as claim 31. Moreover, claim 32 further recites that the SIR includes a summary symbol table. For reasons stated above with respect to claim 5, claim 32 and its depend claims are further allowable over Hiranandani and Ho.

Reversal of the final rejection of the above claims is respectfully requested.

B. Claims 22, 23, and 35 were rejected under 35 U.S.C. § 103(a) as unpatentable over Hiranandani in view of Ho and further in view of Lohmann (U.S. Patent No. 5,826,087).

1. Claims 22, 23, 35.

In view of the allowability of base claims over Hiranandani and Ho, it is respectfully submitted that the obviousness rejection over Hiranandani, Ho, and Lohmann has also been overcome.

Reversal of the final rejection of the above claims is respectfully requested.

C. Claims 15 and 34 were rejected under 35 U.S.C. § 103(a) as unpatentable over Hiranandani in view of Ho and further in view of Haber (U.S. Patent No. 6,966,055).

1. Claims 15, 34.

In view of the allowability of base claims over Hiranandani and Ho, it is respectfully submitted that the obviousness rejection of dependent claims over Hiranandani, Ho, and Haber has been overcome.

Reversal of the final rejection of the above claims is respectfully requested.

CONCLUSION

In view of the foregoing, reversal of all final rejections and allowance of all pending claims is respectfully requested.

Respectfully submitted,

Date: November 6, 2009

/Dan C. Hu/

Dan C. Hu
Registration No. 40,025
TROP, PRUNER & HU, P.C.
1616 South Voss Road, Suite 750
Houston, TX 77057-2631
Telephone: (713) 468-8880
Facsimile: (713) 468-8883

VIII. APPENDIX OF APPEALED CLAIMS

The claims on appeal are:

1. A method of generating a software program executable binary file, the method comprising:

accessing a first file for a first module including source code therein;

accessing a second file for a second module including object code therein and further including object file summary information; and

generating the executable binary file from at least the first and second files,

wherein the object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table, and

wherein the object file summary information is used in optimizing the executable binary file generated.

2. The method of claim 1, further comprising disambiguating memory accesses otherwise considered aliased using the object file summary information.

4. The method of claim 1, wherein the extension to the linker symbol table includes a flag indicating whether a procedure exposes a memory address by storing the address in a location accessible outside the procedure.

5. The method of claim 1, wherein the SIR includes a summary symbol table.

6. The method of claim 5, wherein the summary symbol table includes global and static symbols accessed in a procedure, formal parameters of the procedure, return location for the procedure, and other procedures called by the procedure.

7. The method of claim 6, wherein a symbol is referenced in the summary symbol table by using an associated summary symbol identifier (SYMID).

8. The method of claim 7, wherein a symbol entry includes a linker identifier (LI_ID) of the entry from a linker symbol table.

9. The method of claim 5, wherein the SIR uses an operator for memory referencing.

10. The method of claim 5, wherein the SIR uses an operator to adjust an address expression by an offset.

11. The method of claim 5, wherein the SIR uses an operator to take an address of a function or variable.

12. The method of claim 5, wherein the SIR uses an operator to merge pointer values from different control flow paths.

13. The method of claim 5, wherein the SIR uses an operator to represent direct procedure calls.

14. The method of claim 5, wherein the SIR uses an operator to represent indirect procedure calls.

15. The method of claim 5, wherein the SIR uses a no-operation type operator to discard values.

16. The method of claim 5, wherein the SIR includes a control data structure comprising a link field for each procedure that points to an SIR block of a next procedure.

17. The method of claim 5, wherein the SIR includes a control data structure comprising a table having links to an SIR block for each procedure.

18. The method of claim 1, further comprising determining variables modified by and referenced by function calls in the object code using the object file summary information.

20. The method of claim 18, wherein the extension to the linker symbol table includes a first flag indicative of whether a procedure modifies non-local variables and a second flag indicative of whether the procedure references non-local variables.

21. The method of claim 20, wherein the extension to the linker symbol table includes a second flag indicative of whether the procedure modifies global/static variables excluding callees and a third flag indicative of whether the procedure references non-local variables excluding callees.

22. The method of claim 18, wherein the per-procedure summary data comprises a linked list of entries corresponding to symbols directly modified or referenced in a procedure.

23. (The method of claim 22, wherein each entry comprises a linker identifier of a corresponding symbol and flags indicative of whether that symbol is modified or referenced.

24. The method of claim 1, wherein the second file comprises a load module that is a shared library of procedures.

25. The method of claim 1, wherein multiple files including object code are accessed and used in compiling the program.

26. A system for generating a software program executable file, the system comprising:

a processing device configured to execute computer-readable program code;

a memory system configured to store the computer-readable program code and data;

a source file for a first module comprising source code stored by the memory system;

an object file for a second module including computer-readable program code and object file summary information; and

a translator comprising computer-readable program code stored by the memory system, wherein the computer-readable program code of the translator is configured to access at least the source and object files and to generate the executable file of the program therefrom,

wherein the object file summary information includes a summary intermediate representation (SIR) and an extension to a linker symbol table, and

wherein the object file summary information is used in optimizing the executable file generated.

27. The system of claim 26, further comprising a points-to analyzer that uses the object file summary information to disambiguate memory accesses otherwise considered aliased.

28. The system of claim 26, further comprising a module that uses the object file summary information to determine variables modified by and referenced by function calls in the object file.

29. The system of claim 26, wherein the translator comprises:

a compiler configured to translate source files into intermediate files; and

a linker configured to access the object file summary information and communicate information to the compiler relevant to optimizing compilation of the program.

30. The system of claim 29, wherein the translator further comprises a feedback provider that provides a communications interface between the compiler and the linker.

31. A computer-readable storage medium storing an object file of a computer programming module, the computer-readable storage medium comprising:

computer-readable object code for the module; and

computer-readable object file summary information including a summary intermediate representation (SIR) and an extension to a linker symbol table for use by a compiler in optimizing executable code including the module.

32. The computer-readable storage medium of claim 31, wherein the SIR includes a summary symbol table.

33. The computer-readable storage medium of claim 32, wherein the summary symbol table includes global and static symbols accessed in the module, formal parameters of the module, return location for the module, and other procedures called by the module.

34. The computer-readable storage medium of claim 31, wherein the SIR uses a plurality of operators from a group of operators including an operator for memory referencing, an operator to adjust the address expression by an offset, an operator to take an address of a function or variable, an operator to merge pointer values from different control flow paths, an operator to represent direct procedure calls, an operator to represent indirect procedure calls, .and a no-operation type operator to discard values.

35. The computer-readable storage medium of claim 31, wherein the SIR includes a linked list of entries corresponding to symbols directly modified or referenced in a procedure.

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.